Elmar Schömer
Ann-Christin Wörl

2. Übungsblatt

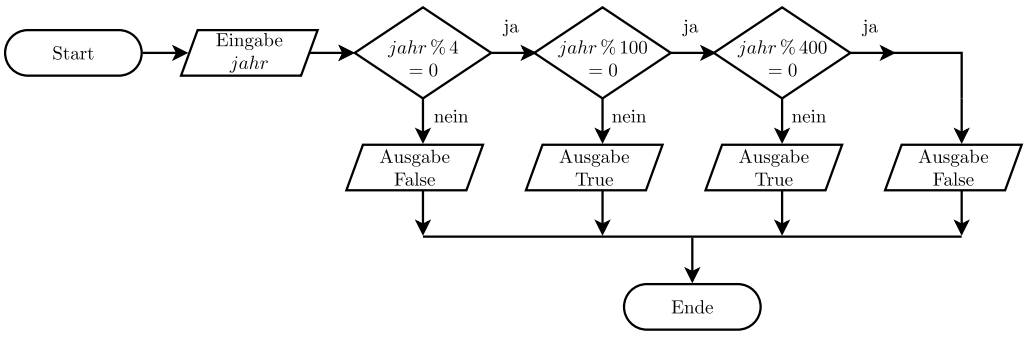
Abgabe: Dienstag, der 07.11.2023, 14:00 Uhr

Aufgabe 1: Schaltjahre (20 Punkt)

Der folgende Programmablaufplan soll dazu dienen, für ein gegebenes Jahr im bei uns geläufigen gregorianischen Kalender zu entscheiden, ob es ein Schaltjahr ist. Dabei gelten folgende Regeln (siehe https://de.wikipedia.org/wiki/Schaltjahr):

- Die durch 4 ganzzahlig teilbaren Jahre sind Schaltjahre, mit Ausnhame von:
- Säkularjahre, also die Jahre, die ein Jahrhundert abschließen, sind keine Schaltjahre, außer:
- Die durch 400 ganzzahlig teilbaren Säkularjahre sind jedoch Schaltjahre.

Erklärung der Notation: $n \,\%\, m$ (oft auch $n \bmod m$) steht für den ganzahligen Rest bei der Division von n durch m, d.h. wenn $n = q \cdot m + r$ mit $0 \leq r < m$, dann bezeichnet $r = n \,\%\, m$ den Rest, wobei $n, m, q, r \in \mathbb{N}$, also positive Ganzzahlen sind.



1. Ist der obige Programmablaufplan korrekt? Begründen Sie Ihre Antwort, und beschreiben Sie gegebenenfalls, wo ein Fehler vorliegt.
2. Waren die Jahre 2000, 2020 und 2022 Schaltjahre?
3. Schreiben Sie ein Python-Programm, das den obigen (ggfs. korrigierten) Programmablaufplan umsetzt.

```
1  print(2022 // 4) # ganzzahlige Division
2  print(2022 % 4)  # Rest der ganzzahligen Division
```

Aufgabe 2: Zinseszins (20 Punkt)

Wir betrachten noch einmal die Zinseszinsformel:

$$K = K_0 \cdot (1 + p)^n$$

1. Schreiben Sie ein Python-Programm, das die Laufzeit n in Jahren berechnet, wenn die Werte für das Startkapital K_0, das Endkapital K und der Zinssatz p bekannt sind. Wenn Sie eine besondere mathematische Funktion benötigen, finden Sie die Dokumentation dazu unter https://docs.python.org/3/library/math.html.

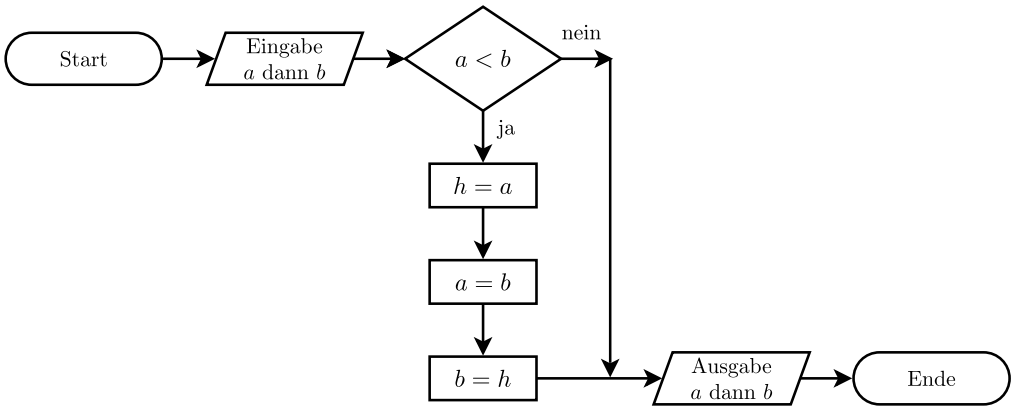
```
1  import math
2
3  K0 = float(input('K0 ='))
4  K  = float(input('K  ='))
5  p  = float(input('p  ='))
6
7  print('n = ')
```

2. Schreiben Sie ein Python-Programm, das den Zinssatz p berechnet, wenn die Werte für das Startkapital K_0, das Endkapital K und die Laufzeit n in Jahren bekannt sind.

```
1  import math
2
3  K0 = float(input('K0 ='))
4  K  = float(input('K  ='))
5  n  = float(input('n  ='))
6
7  print('p = ')
```

Aufgabe 3: Sortieren (20 Punkt)

Der folgende Programmablaufplan dient dazu, zwei ganze Zahlen a und b zu sortieren. Der Wert der zuerst eingelesenen Zahl wird in der Variable a gespeichert und der Wert der zweiten Zahl in der Variable b. Falls $a < b$, werden die Werte der Variablen von a und b unter Zuhilfenahme einer Variablen h vertauscht. Abschließend wird zuerst $\max(a, b)$ und dann $\min(a, b)$ ausgegeben.



1. Ergänzen Sie das folgende Programm, so dass es zwei Zahlen absteigend sortiert.

```
1  a = int(input('a = '))
2  b = int(input('b = '))
3
4  print(a,b)
```

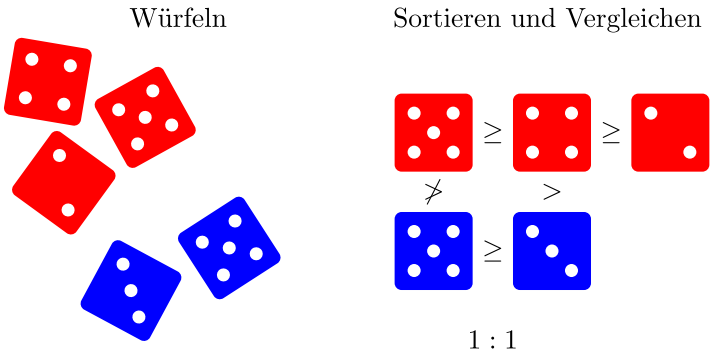
2. Ergänzen Sie das folgende Programm, so dass es drei Zahlen absteigend sortiert.

```
1  a = int(input('a = '))
2  b = int(input('b = '))
3  c = int(input('c = '))
4
5  print(a,b,c)
```

Aufgabe 4: Risiko (20 Punkt)

Bei dem Brettspiel Risiko (siehe https://de.wikipedia.org/wiki/Risiko_(Spiel)) tritt ein Angreifer mit drei roten Würfeln gegen einen Verteidiger mit zwei blauen Würfeln an. Alle Würfel werden gleichzeitig geworfen. Dann wird die höchste Augenzahl r der roten Würfel mit der höchsten Augenzahl b der blauen Würfel verglichen. Wenn $r > b$, dann gewinnt der Angreifer einen Punkt, ansonsten der Verteidiger. Ebenso wird mit den zweithöchsten Augenzahlen verfahren (siehe Abb.). Es sind also drei Endergebnisse möglich:

$$2:0, 1:1 \text{ oder } 0:2$$



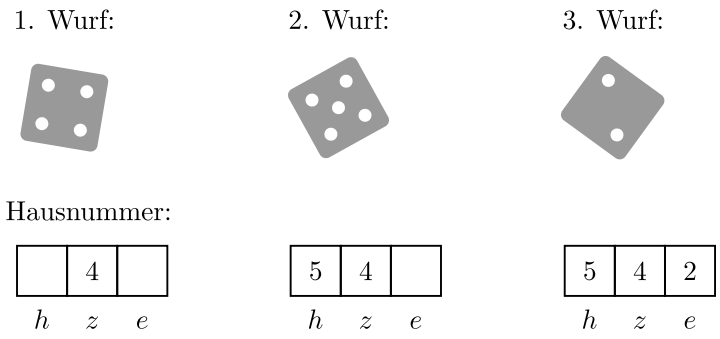
Die Funktion `random.randint(1,6)` simuliert einen fairen Würfel. Schreiben Sie ein Python-Programm, das für jedes Würfelereignis den Punktestand "Angreifer : Verteidiger" ermitteln kann.

```
1  import random
2
3  r1 = random.randint(1,6)
4  r2 = random.randint(1,6)
5  r3 = random.randint(1,6)
6  print(r1,r2,r3)
7
8  b1 = random.randint(1,6)
9  b2 = random.randint(1,6)
10 print(b1,b2)
```

Freiwillige Aufgabe für all diejenigen, die eine zusätzliche Herausforderung suchen:

Zusatzaufgabe 5: Hohe Hausnummer (20*+10*+10* Punkt)

Beim Würfelspiel "Hohe Hausnummer" (siehe https://de.wikipedia.org/wiki/Hohe_Hausnummer) würfelt man drei mal hintereinander mit einem Würfel. Mit den drei gewürfelten Augenzahlen soll eine möglichst große dreistellige Zahl, die Hausnummer, gebildet werden. Bereits nach dem 1. Wurf muss man entscheiden, ob die geworfene Augenzahl auf die Einer-, die Zehner- oder die Hunderterstelle gesetzt werden soll. Und auch nach dem 2. Wurf muss man festlegen, auf welche noch freie Stelle diese Augenzahl gesetzt werden soll. Die Augenzahl des 3. Wurfes wird dann auf die verbliebene freie Stelle gesetzt.
Wir spielen dieses Spiel allein und fragen uns, mit welcher Strategie wir möglichst hohe Hausnummern erzielen können. Im folgenden Beispiel haben wir gute Entscheidungen nach den einzelnen Würfen getroffen, so dass wir die höchst mögliche Hausnummer für die drei gegebenen Augenzahlen bilden konnten. Das ist natürlich nicht immer möglich, weil wir z.B. nicht wissen können, ob nicht doch noch im letzten Wurf eine 6 fällt.



1. Das folgende Programm benutzt eine schlechte Strategie zum Bilden der Hausnummer. Versuchen Sie, eine bessere Strategie zu programmieren!

```
1  import random
2
3  w1 = random.randint(1,6)
4  h = w1
5  w2 = random.randint(1,6)
6  z = w2
7  w3 = random.randint(1,6)
8  e = w3
9  hausnummer = 100*h+10*z+e
10
11 print(hausnummer)
```

2. Geben Sie eine optimale Strategie an, die über viele Würfelexperimente hinweg (im Erwartungswert) die höchst mögliche Hausnummer für einen Solo-Spieler verspricht.

3. Jetzt treten zwei Spieler gegeneinander an, ohne den Spielverlauf des jeweils anderen Spielers beobachten zu können. Der Spieler mit der höheren Hausnummer gewinnt. Gibt es eine Strategie, mit der man auf lange Sicht gegen die optimale Solo-Strategie gewinnen kann?